Core Animation Techniques for iPhone and Mac

Session 303

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

Core Animation Techniques for iPhone and Mac

John Harper

Agenda

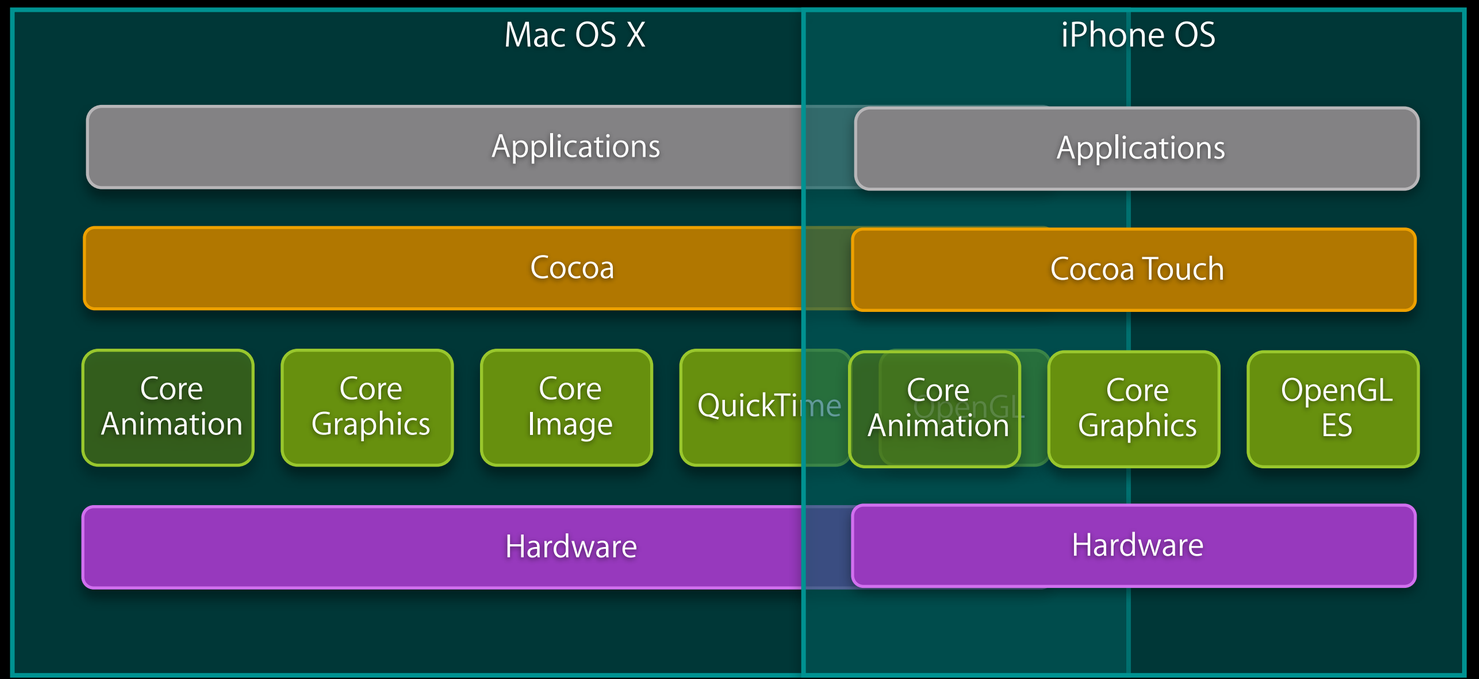
- Background
- Techniques
- Performance

Background

Layer Compositing

- Foundation of most 2D graphics
- Want expressiveness of professional design applications in mobile and desktop user interfaces
- Your user interface is a tree of “layers”
- Layers have geometry, content, filters, etc
- Attach animations to layers to add motion/transitions

Frameworks

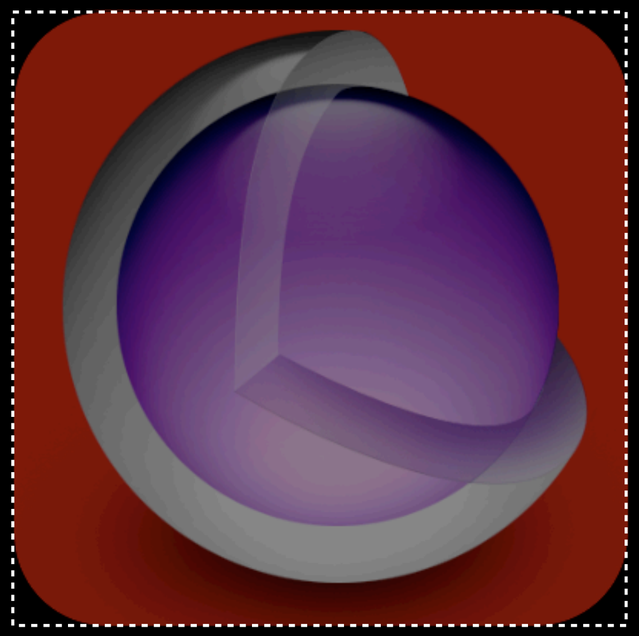

Mac OS X
iPhone OS
Applications
Applications
Cocoa
Cocoa Touch
Core Animation
Core Graphics
Core Image
QuickTime
Core Animation
OpenGL
Core Graphics
OpenGL ES
Hardware
Hardware

When to Use Core Animation

- Use high-level frameworks when possible
  - Events, text, accessibility, etc. all handled for you
  - Views will implicitly create layers
- When these limit you, use Core Animation directly
  - Add custom animations to views
  - Add custom layers to view tree
- Use Core Animation entirely for “immersive” interfaces

Basics

Getting Started



```
CALayer *l = [CALayer layer];
l.frame = CGRectMake (0, 0, 300, 300)];
[parent addSublayer:layer];
l.backgroundColor = [[UIColor redColor] CGColor];
l.contents = [[UIImage imageWithContentsOfFile:…]
 CGImage];
l.cornerRadius = 40;
l.opacity = 0.5;
```

Custom Drawing

- Define your drawing method
  - By subclassing CALayer

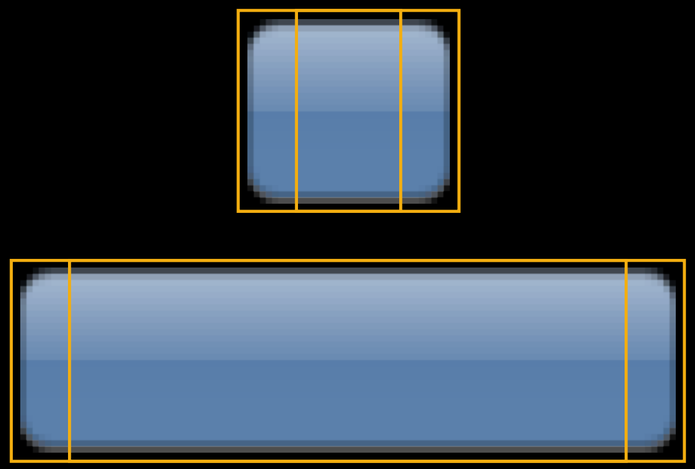
```
-(void)drawInContext:(CGContextRef)ctx {
    // draw something into ctx…
```

  - Or by setting a delegate object on the layer

```
-(void)drawLayer:(CALayer*)l inContext:(CGContextRef)ctx {
    // draw something into ctx…
```

- Call -setNeedsDisplay or -setNeedsDisplayInRect:
- Read context's clip rect to optimize drawing
- Set the layer's opaque property to YES if alpha not needed

9-Part Images



```
layer.contentsCenter = CGRectMake (.2, 0, .6, 1);
layer.frame = CGRectInset (layer.frame, -100, 0);
```

- 3- and 9-part images common in UI
- Center piece scales with layer
- Makes animating layer size much easier
- Can also crop images using contentsRect

Tiled Images

- Large images are now implicitly tiled by Core Animation
- For explicit control of tiling, draw into a CATiledLayer
  - No extra copies of the image data; optimal tile caching
  - Supports multiresolution image data
  - Redefines layer semantics to be asynchronous
- Pretile into many small image files for best drawing performance

Animations

Property Changes

- Almost all property changes implicitly animate

```
layer.opacity = 0.5;  // adds a quarter second animation
```

- Disable implicit animations entirely

```
[CATransaction setDisableActions:YES];
```

- Change the duration

```
[CATransaction setAnimationDuration:1];
```

- Or the timing function

```
[CATransaction setAnimationTimingFunction:[CAMediaTimingFunction
 functionWithName:kCAMediaTimingFunctionEaseIn]];
```

“Flip-Book” Animations

```
CGImageRef im[N]; load_images (im, N);
CAKeyframeAnimation *anim = [CAKeyframeAnimation animation];
anim.keyPath = @”contents”;
anim.duration = 1;   // frame rate = N / duration
anim.calculationMode = kCAAnimationDiscrete;
anim.repeatCount = HUGE_VAL;
anim.values = [NSArray arrayWithObjects:im[0], im[1], …, nil];
[layer addAnimation:anim forKey:@”spinner”];
```

- Just one example of using explicit animations
  - Blink cursors using opacity animations
  - Wherever no natural state transition occurs

Transitions

- Some state changes can’t be animated as property values
- For these, CATransition animation class
- Several built-in types; Mac can use any Core Image transition
- “from” state is the previous render tree; “to” is current render tree
- You will sometimes want to disable the implicit transitions

```
- (id<CAAction>)actionForKey:(NSString*)key {
    if ([key isEqualToString:@”sublayers”]) return nil;
    else return [super actionForKey:key];
}
```

- Note: Delegate method should return [NSNull null]!

Notifications

- Animations created explicitly can use a delegate object
- Implicit animations can use “completion block” (Mac only)

```
[CATransaction begin];
[CATransaction setCompletionBlock:^{
    // block that runs when animations have completed
    [CATransaction setDisableActions:YES];
    [layer removeFromSuperlayer];
}];
layer.opacity = 0;
layer.position = CGPointMake (2000, layer.position.y);
[CATransaction commit];
```

Querying Values

- Layer properties do not reflect active animations
- Use -presentationLayer method to get “screen” values
  - Creates a temporary layer with animations applied
  - Asking for sublayers returns presentation versions
- Useful for “from” values of animations

```
anim = [CABasicAnimation animationWithKeyPath:@”borderColor”];
anim.fromValue = [[layer presentationLayer] borderColor];
```

- And for hit-testing against real geometry

```
hitLayer = [[[layer presentationLayer] hitTest:p] modelLayer];
```

Animated Drawing

- Properties defined in subclasses can now be animated

```
@property CGFloat lineWidth;
```

- Mark it dynamic so that CALayer implements the accessors

```
@dynamic lineWidth;
```

- Mark that changing the property should redraw the layer

```
- (BOOL)needsDisplayForKey:(NSString*)key {
    if ([key isEqualToString:@”lineWidth”]) return YES;
    else return [super needsDisplayForKey:key];
}
```

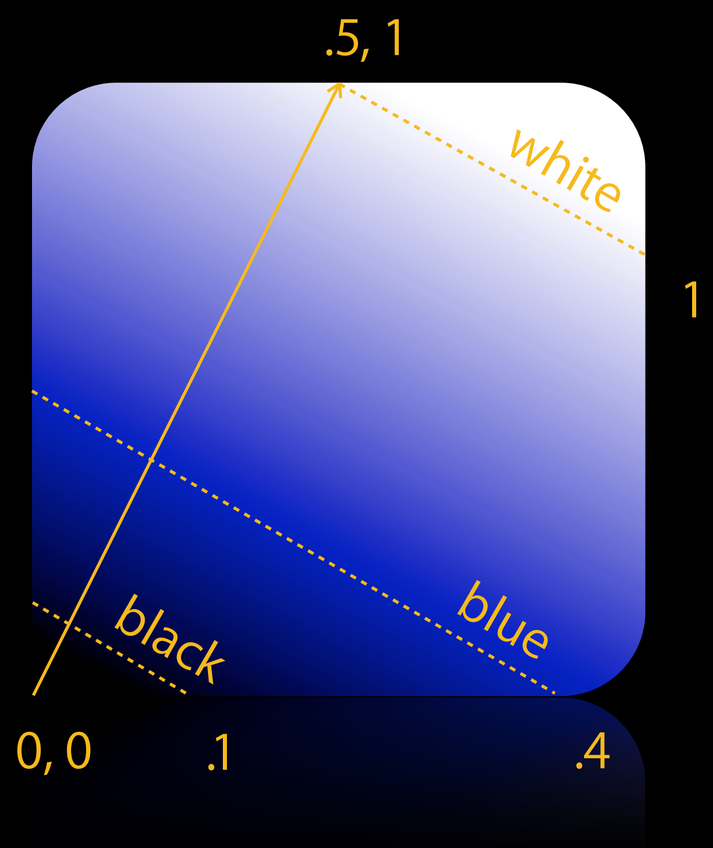
- Use the property’s value in the -drawInContext: method
- For implicit animations, override -actionForKey: method

Animated Drawing

- How does this work?
  - Animations on properties start a periodic timer
  - Values are applied to a copy of the layer every frame
  - -drawInContext: is sent to the copy of the layer
- What types can be interpolated?
  - Numbers, geometry values, colors, paths, arrays of these types
- Caveats
  - Only works for @property's
  - Evaluation times are approximate

Content Types

Gradients




```
 CAGradientLayer *layer = [CAGradientLayer layer];
 layer.type = kCAGradientLayerAxial;
 layer.startPoint = CGPointMake (0, 0);
layer.endPoint = CGPointMake (.5, 1);
 CGColorRef black = …, blue = …, white = …;
layer.colors = [NSArray arrayWithObjects:black,
 blue, white, nil];
layer.locations = [NSArray arrayWithObjects:
 [NSNumber numberWithFloat:.1], [NSNumber
 numberWithFloat:.4], [NSNumber numberWithInt:1],
 nil];
```

Text

hello, world

```
CATextLayer *layer = [CATextLayer layer];
layer.string = @”hello, world”;
layer.font = @”Helvetica”;
layer.fontSize = 36;
```

- String can be NSString or NSAttributedString
- Font can be CGFontRef, CTFontRef or font name
- Can also enable line-wrapping

Text

hello, world

hello, world

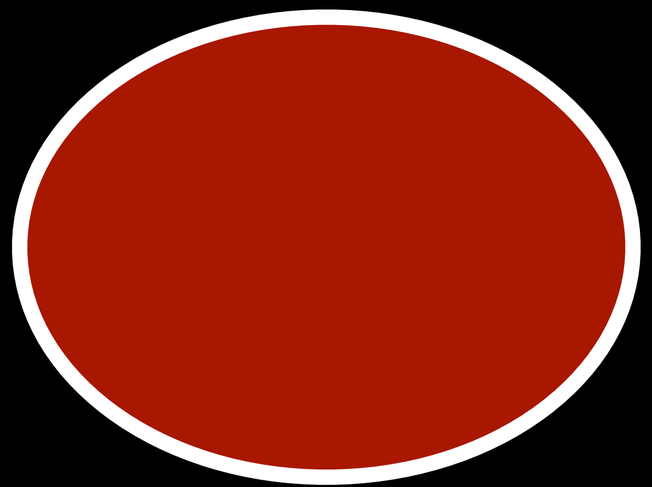
hello, world

```
@interface MyTextLayer : CATextLayer @end

@implementation MyTextLayer
- (void)drawInContext:(CGContextRef)ctx {
  // Fill opaque background and enable smoothing
  CGContextSetRGBFillColor(ctx, .4, .6, .8, 1);
  CGContextFillRect (ctx, [self bounds]);
  CGContextSetShouldSmoothFonts (ctx, true);
  // Enable shadows
  CGContextSetShadow (ctx, CGSizeMake (0,-4), 2);
  [super drawInContext:ctx];
}
@end
```

Bézier Paths

Bézier Paths



```
CGRect r = CGRectMake (0, 0, 400, 300);
CGMutablePathRef path = CGPathCreateMutable ();
CGPathAddEllipseInRect (path, NULL, r);

CAShapeLayer *layer = [CAShapeLayer layer];
layer.path = path;
layer.lineWidth = 10;
layer.strokeColor = white_CGColor ();
layer.fillColor = red_CGColor ();

CGPathRelease (path);
```

Bézier Paths

- Possible uses
  - Shape elements, e.g. arrows, stars, speech bubbles, etc.
  - “marching ants” (two layers same path, one dashed, animate phase)
  - Text (layer per glyph, reuse path objects for same glyph)
  - SVG-style images
  - As a mask for another layer

Animating Paths

- Now we can display CGPath objects, can we animate them?
- Yes, BUT
  - The two paths must have the same structure for acceptable results
  - CAShapeLayer path property does not implicitly animate

Other Content Types

	Mac OS X	iPhone OS
OpenGL	CAOpenGLLayer	CAEAGLLayer
QuickTime	QTMovieLayer QTCaptureLayer	
Quartz Composer	QCCompositionLayer	

Special Effects

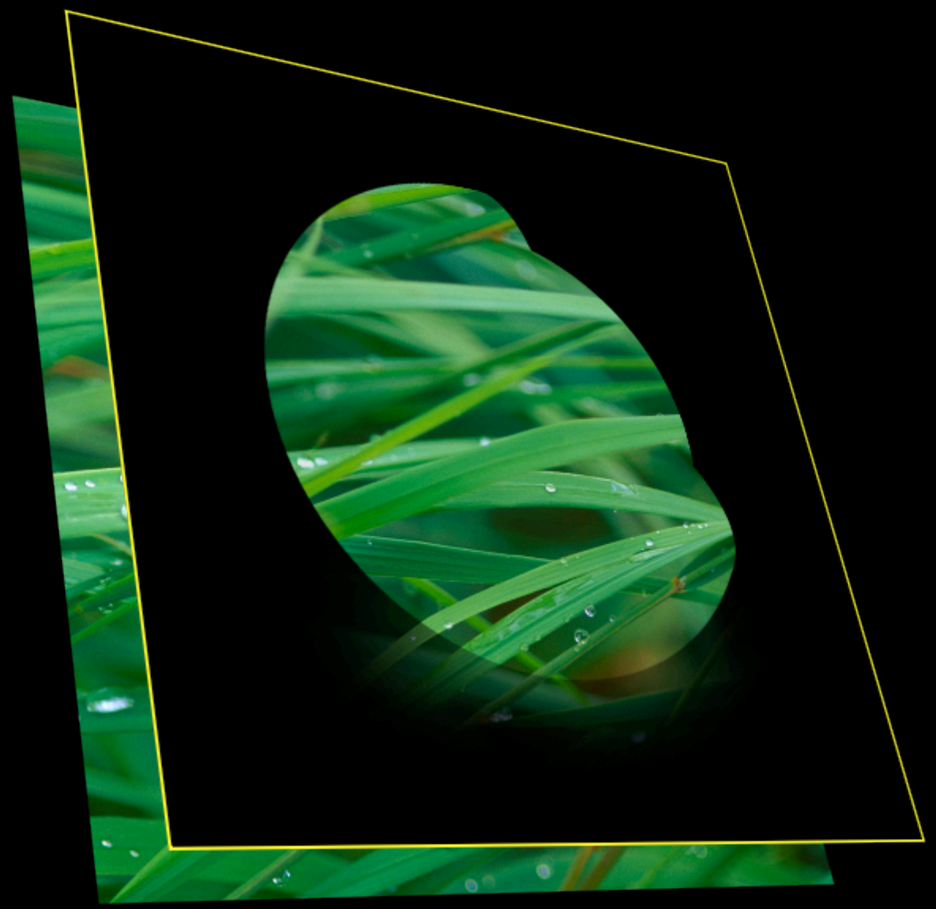
Masks

```
CALayer *bg = grass_blades_layer ();
CAlayer *fg = pink_ball_layer ();

bg.mask = fg;
bg.sublayers = [NSArray arrayWithObject:fg];
```

- Each layer may have a mask layer
- Mixes composited layer and backdrop
- Acts like any other sublayer
- Applying mask may be expensive…

Masks



- …but you can often “cheat”
- Requires known background
- (¬mask × bg) over fg
- So for soft scroller edges add two gradient layers

Filters

```
CIFilter *filter = [CIFilter filterWithName:@”CIBloom”];
[filter setDefaults];
[filter setName:@”bloom1”];  // name is used in key-path
layer.filters = [NSArray arrayWithObject:filter];
```

- Inputs of named filters can be set and animated

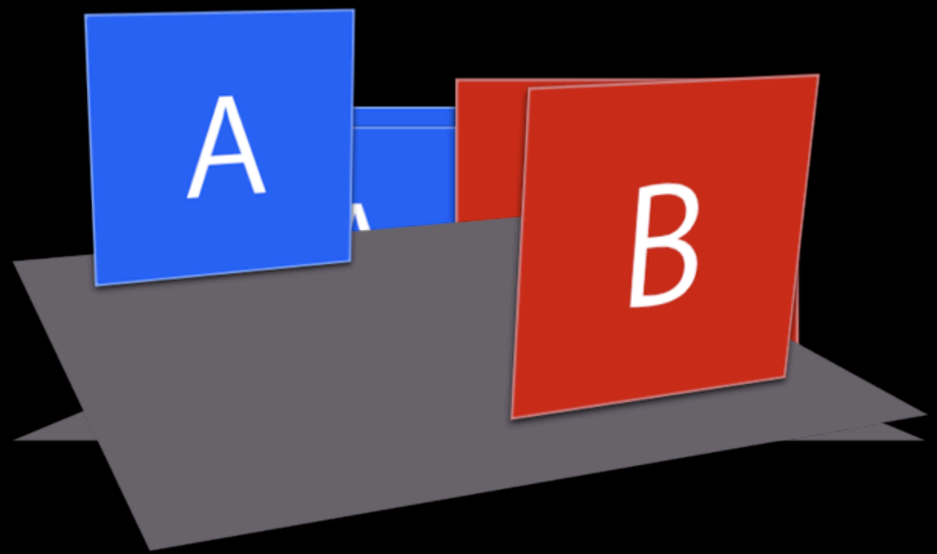
```
[layer setValue:[NSNumber numberWithInt:20]
 forKeyPath:@”filters.bloom1.inputRadius”];
```

- For static filters consider burning the filter into the layer’s drawn content—often far cheaper

Background and Compositing Filters

```
CALayer *bg = grass_blades_layer ();
CAlayer *fg = pink_ball_layer ();
bg.sublayers = [NSArray arrayWithObject:fg];
CIFilter *bgfilter = [CIFilter
 filterWithName:@”CICrystallize”];
[bgfilter setDefaults];
fg.backgroundFilters = [NSArray
 arrayWithObject:bgfilter];
fg.compositingFilter = [CIFilter
 filterWithName:@”CIHardLightBlendMode”];
fg.mask = pink_ball_layer ();
```

2½D—Perspective



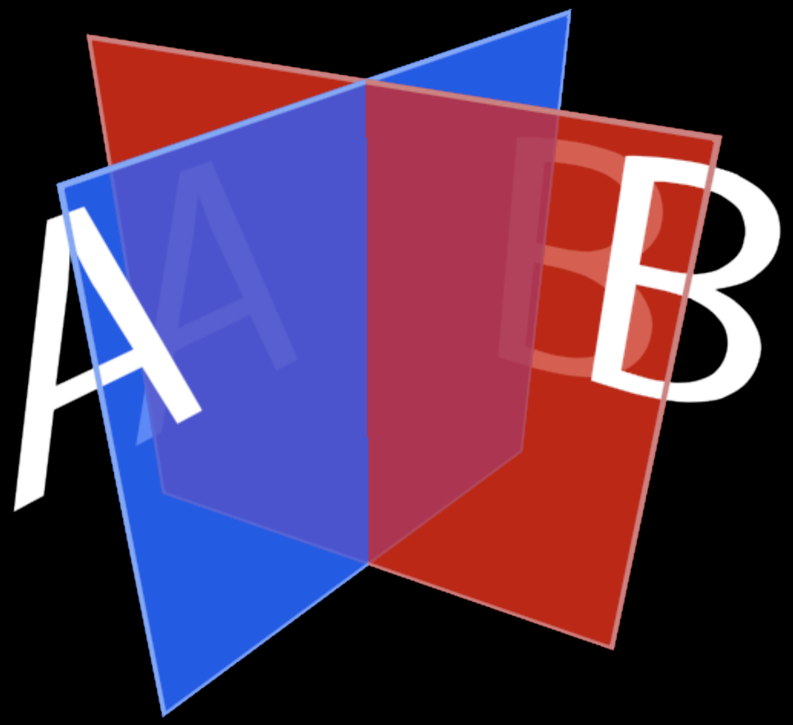

```
CALayer *a = blue_square_layer ();
CALayer *b = red_square_layer ();
p.sublayers = [NSArray arrayWithObjects:a,b,nil];
a.zPosition = -100;
b.zPosition = 100;

CATransform3D t = CATransform3DIdentity;
t.m34 = -1. / EYE_Z;
p.sublayerTransform = CATransform3DConcat(rotationMatrix, t);
t = CATransform3DTranslate (t, -cX, -cY, -cZ);
```

2½D—Groups

- Layers are "2½ D", flattened into parent's Z=0 plane
- Scenes with depth must be a flat list of layers, no groups
- Use CATransformLayer to create "3D groups"
  - Geometry and opacity is applied to each sublayer
  - Sublayers are hoisted into parent and sorted
- Transform layers have no content themselves
  - No image, no color, no filters, no masking, no compositing

2½D—Depth Sorting



- Intersecting layers are supported
- But best avoided
  - Non-trivial extra work for renderer
  - Layers are rendered more than once
- Depth-sorting uses layer bounds and position to determine occlusion

Drop Shadows

- Three methods
  1. Use CALayer shadow properties (Mac only)
  2. Draw shadow into layer bitmap
  3. Use constant shadow image in separate layer(s)
- Built-in shadows are expensive—per-frame blur operation
- Drawing shadow in layer works well for text
- Can use 9-part image for shadows on opaque layers

Reflections

```
CAReplicatorLayer *l = [CAReplicatorLayer layer];
l.sublayers = layers_to_be_reflected ();
l.instanceCount = 2;
CATransform3D t = CATransform3DIdentity;
t = CATransform3DScale (t, 1, -1, 1);
t = CATransform3DTranslate(t, 0, H - 2*rY, 0);
l.instanceTransform = t;
l.instance{Red,Green,Blue}Offset = -.75;
```

Particles

- Useful for UI flourishes—CAEmitterLayer
- Layer has an emission shape and an array of cells
- Cells define direction and other per-particle properties
- Cells can have subcells—particles that emit particles
- Emitter layers can be 2D or 3D

Performance

Performance Rules

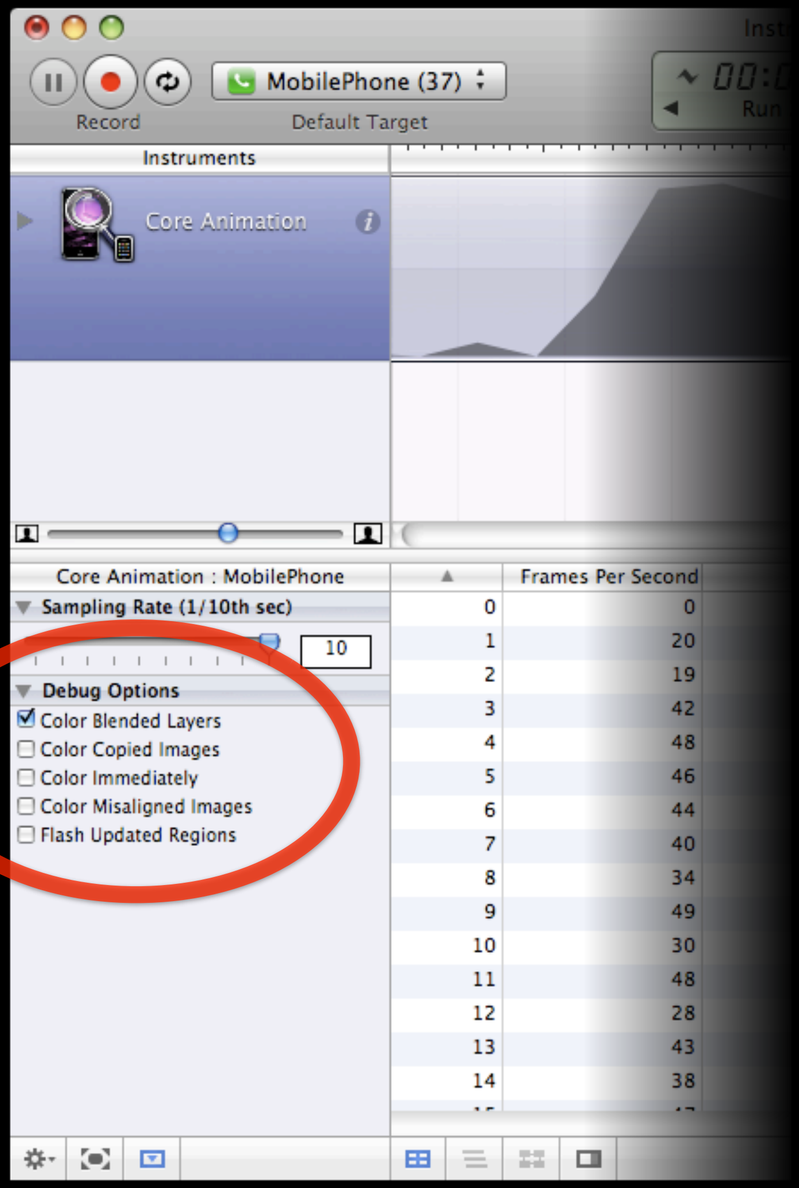
1. Avoid offscreen rendering

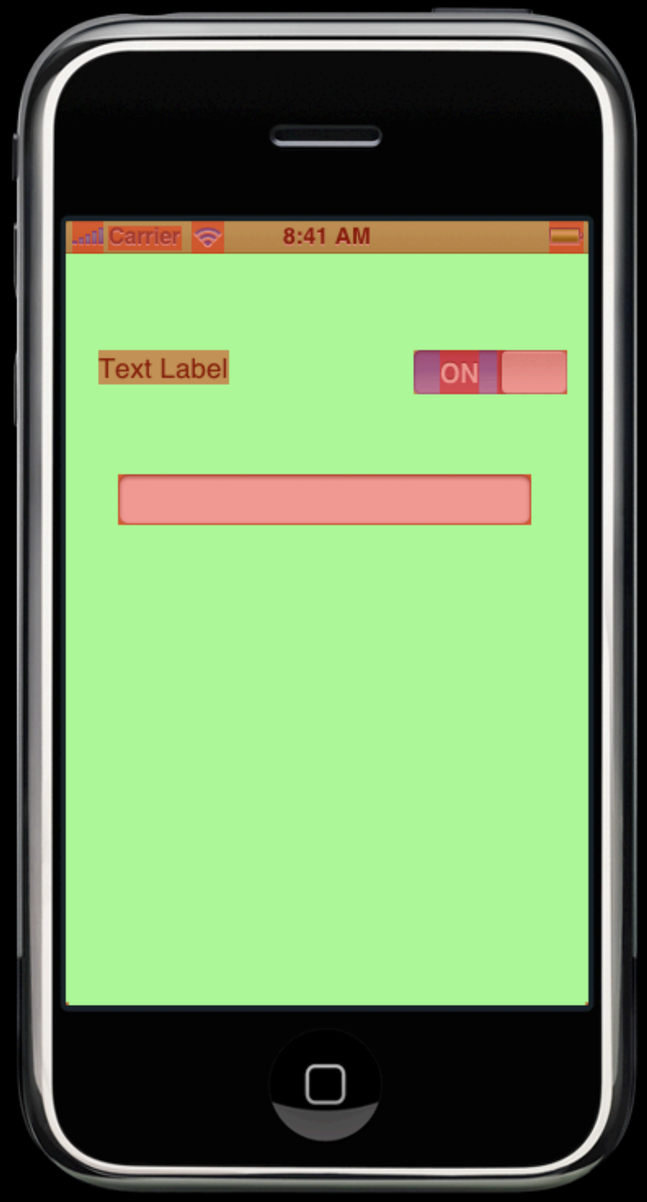
- Changing opacity of layers with >1 color source
- Mask layers, or masksToBounds when not pixel-aligned
- Core Image filters, shadows (Mac only)
- Transition effects
- Nested 3D projections

Performance Rules

2. Minimize alpha-blending

- The key to iPhone UI performance
- If drawing into a layer: layer.opaque = YES
  - If unable to do this, consider collapsing layer subtree
- If providing a CGImageRef: Ensure it has no alpha channel
  - Fix your image files and use kCGImageAlphaNone when creating from data
- On Mac: Environment variable CA_COLOR_OPAQUE=1
- On iPhone: Color blended layers option in Instruments


Record
MobilePhone (37)
Default Target
Instruments
Core Animation
Core Animation : MobilePhone
Frames Per Second
Sampling Rate (1/10th sec)
10
Debug Options
Color Blended Layers
Color Copied Images
Color Immediately
Color Misaligned Images
Flash Updated Regions
0 0
1 20
2 19
3 42
4 48
5 46
6 44
7 40
8 34
9 49
10 30
11 48
12 28
13 43
14 38


Carrier
8:41 AM
Text Label
ON

Wrapping Up

Summary

- UI frameworks create Core Animation layers for every view
- Try to use these APIs whenever possible, makes your life easier
- Minimize amount of translucency in iPhone apps

Final Thoughts

See us in the lab this afternoon!

Allan Schaffer
Graphics Technology Evangelist
aschaffer@apple.com

Mailing lists

quartz-dev@lists.apple.com
cocoa-dev@lists.apple.com

Q&A

WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09